Data fusion and Bayesian Networks

Prof. C. Regazzoni
Data Fusion and Bayesian Interaction Modeling for Cognitive Ambient Intelligence
Outline

• Introduction to Data Fusion
  ▪ Motivations
  ▪ Input and Output of a Data Fusion System
  ▪ Application Domains
  ▪ Data Fusion Process Models

• Functional Models
  ▪ JDL model
  ▪ Levels of the JDL model
  ▪ Other Functional Models

• Architectural Models
  ▪ Centralized Models
  ▪ Distributed Models
Outline

• **Mathematical Models**
  - Processing in Data Fusion Levels
  - Data Preparation
    - Temporal Alignment - NTP
    - Spatial Alignment – Tsai’s Calibration
  - Data Association
    - General Remarks
    - Nearest Neighbour
    - PDAF and JPDAF
    - Graph-based Data Association
Outline

- **State Estimation**
  - Bayesian Filtering
  - Kalman Filter
  - Extended and Unscented Kalman Filter
  - Particle Filter
  - PHD Filter
- **Identity Declaration**
  - Bayesian Decision
  - Artificial Neural Networks
- **Distributed Data Fusion**
  - Distributed State Estimation
    - Distributed Kalman Filter
  - Distributed Detection
    - Detection with Parallel Fusion Networks
Introduction
**Introduction**

**Multisensor data fusion** techniques jointly compute information gathered by multiple and, eventually, heterogeneous sensors to perform inference not obtainable with a single sensor.

- Homogeneous sensors
- Heterogeneous sensors

Visible

Infrared
Data fusion in biological systems

- Multisensory data fusion resembles the **cognitive process** by animals and humans: *data streams* coming from senses are jointly analysed to provide a common context representation that improves the chances of survival of the organism.
- Sights, sounds, smells, tastes and touch are features assessed by the human brain for recognition and situation evaluation.
- Many of the techniques developed for data fusion attempt to emulate this ability.
What is Data Fusion?

• The Data Fusion problem consists in achieving a consistent, comprehensive estimate and prediction of some relevant portion of the world state.

• Data Fusion involves exploiting all sources of data available (inputs) to perform all relevant state estimation and prediction, where relevance consists in the capability of forming plans of action (outputs).

• Data Fusion is then a process of combining data or information to estimate or predict entity states. The estimates and assessments are incremental and the evaluation of the need for additional sources or process modification is performed to improve the quality of the output.
Inputs of a Data Fusion system

The inputs of a sensor Data Fusion system is composed by three basic elements:

- data observed by sensors
- data and commands inputs by human operators or users
- a-priori data from a pre-established database

Inputs data can be categorized according to the energy spectrum of the sensors.
## Inputs of a Data Fusion system

<table>
<thead>
<tr>
<th>Sensors</th>
<th>Frequency</th>
<th>Spectral range</th>
</tr>
</thead>
<tbody>
<tr>
<td>UV spectrometers</td>
<td>10 MHz – 100 MHz</td>
<td>200 – 380 nm</td>
</tr>
<tr>
<td>Visible Cameras</td>
<td></td>
<td>780 – 380 nm</td>
</tr>
<tr>
<td>Infrared Cameras</td>
<td></td>
<td>3 nm – 1 m</td>
</tr>
<tr>
<td>Radars</td>
<td></td>
<td>10 MHz – 100 MHz</td>
</tr>
<tr>
<td>Acoustic sensors</td>
<td>1 Hz – 10 KHz</td>
<td>200 – 380 nm</td>
</tr>
</tbody>
</table>
The output of a data fusion system should be intended to support human decision process and not simply to process efficiently the acquired sensor data.
Output of a Data Fusion system

The output of a Data Fusion system can be provided at different **level of inference** to provide a **hierarchical** estimate of the state.

**Types of inference**
- Threat analysis
- Situation assessment
- Behavior of entity
- Identity
- Position velocity
- Existence of an entity

**Types of data/processing**
- Contextual analysis
- Analysis of Mission, goal, situation
- Time and local geometric analysis
- Parametric data
- Multiple data for position estimation
- Raw sensor data
Evolution of data fusion technology:

1. improved **collection** of related techniques

2. well founded discipline with standardized **terminology**, robust mathematical techniques and established system design principles.

Many **disciplines** are involved in the development of data fusion systems:

- signal processing
- pattern recognition
- control theory
- artificial intelligence
- information theory
- statistical estimation
- cognitive psychology
Applications domain

Data fusion methods, initially developed for military purposes, are nowadays widely exploited in civilian applications.

<table>
<thead>
<tr>
<th>Application Examples</th>
<th>Inference sought</th>
<th>Primary observable data</th>
<th>Surveillance volume</th>
<th>Sensor Platforms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strategic warning and defence</td>
<td>Detection of indications of impending strategic actions, detection and tracking of ballistic missiles and warheads</td>
<td>EM radiation, nuclear particles</td>
<td>Global</td>
<td>Satellites, aircrafts</td>
</tr>
<tr>
<td>Assets surveillance</td>
<td>Detection, tracking, identification of targets and events</td>
<td>EM radiation, acoustic emissions, vibrations, temperature and pressure</td>
<td>Buildings, Airports, Blocks, Towns</td>
<td>Buildings, vehicles, aircrafts</td>
</tr>
<tr>
<td>Air-to-air and surface-to-air defence</td>
<td>Detection, tracking, identification of aircrafts</td>
<td>EM radiation</td>
<td>Hundreds of miles (strategic), miles (tactical)</td>
<td>Aircraft, ground-based</td>
</tr>
<tr>
<td>Battlefield intelligence, surveillance, and target acquisition</td>
<td>Detection of indications of potential ground targets</td>
<td>EM radiation</td>
<td>Tens of hundreds of miles about a battlefield</td>
<td></td>
</tr>
</tbody>
</table>
# Applications domain/2

<table>
<thead>
<tr>
<th>Application</th>
<th>Inference sought</th>
<th>Primary observable data</th>
<th>Surveillance volume</th>
<th>Sensor Platforms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environmental monitoring</td>
<td>Identification and location of natural phenomena (e.g.: earthquakes, fires, weather, …)</td>
<td>SAR, seismic, EM radiation, core samples, chemical and biological data</td>
<td>Hundreds of square km</td>
<td>Satellites, aircrafts, ground stations, underground samples</td>
</tr>
<tr>
<td>Robotics</td>
<td>Object location &amp; recognition, movement</td>
<td>Television, acoustic signals, EM signals</td>
<td>Near-location about the robot</td>
<td>Robot body</td>
</tr>
<tr>
<td>Medical diagnosis</td>
<td>Diagnosis of diseases, tumors and physical conditions</td>
<td>X-rays, NMR, Temperature, IR, visual inspection, acoustic emissions, chemical and biological samples</td>
<td>Volume of human body or observed part</td>
<td>Laboratory</td>
</tr>
<tr>
<td>Automated monitoring of equipment</td>
<td>Satus and health of equipment, identification of impending fault conditions</td>
<td>EM radiation, acoustic emissions, vibrations, temperature and pressure</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Advantages of Data Fusion

• Increase of reliability
• Decrease of ambiguity; the combination of more information reduce the uncertainty
• Increase of detection performances
• Increase of robustness; a device can substitute a damaged one
• Enlarged time and spatial coverage
• Decrease of costs; more simple devices can substituted one more complex and expensive
Limitations

- Good sensors are still needed: a single accurate source of measurement cannot be substituted by any amount of data fusion
- Downstream processing cannot make up for errors in upstream processing
- Poor performance with incorrect information about sensors performance
- No silver bullet algorithm
- Need of a large amount of training data
- Fusion is not a static process. Then methods to perform adaptation of the algorithms is necessary
- The evaluation of a data fusion system is a difficult process
The data fusion process can be described under different points of view:

- **functional model**, to describe the primary functions, relevant databases and interconnectivity
- **architectural model**, to specify hardware and software building blocks, associated data flows and external interfaces
- **mathematical model**, to describe algorithms and logical processes

The rest of the presentation will describe these three aspects of the data fusion problem.
Functional Models
Between the different proposal of functional models, the **JDL (Joint Directors of Laboratories) Process Model** is one of the mostly used.

**JDL (Joint Directors of Laboratories) Process Model:**

- Source Pre-Processing
- Level One Object Refinement
- Level Two Situation Refinement
- Level Three Threat Refinement
- Level Four Process Refinement
- Level Five Cognitive Refinement
- Database Management System
  - Support Database
  - Fusion Database
- Human Computer Interaction
Different **functional models** based on JDL terminology have been proposed during these years. In the late 90s the model was **updated** [1] reaching the current version.

JDL Process Model (Pre-processing)

**Preliminary filtering** on data from sensors is needed to standardize inputs, correct biases, remove noise, extract key information.

- Examples of source pre-processing include image processing, signal processing, noise filtering.
- An important task that belongs to this stage is the **feature extraction**. Relevant information must be extracted from raw data excluding the not interesting ones for the fusion process. For example, the contour detection in images allows to provide shape descriptions.
- They are useful to render the following Data Fusion processes more effectiveness.
JDL Process Model (Level 1)

**Level one (object refinement)** processing fuses filtered sensors data to determine the identity, position and velocity of entities.

Four functions can be distinguished:

- **data alignment**: it transforms data received from multiple sensors into a common spatio-temporal reference frame
- **data association**: it sorts or correlate observations from multiple sensors into different groups representing data related to a single distinct entity
- **tracking**: it aims to determine an estimate of targets position and speed from multiple observations of positional data coherently for successive instants
- **identification**: it classifies the objects originated by the measurements
Level two (situation refinement) processing performs an improved degree of inference with respect to level one. To accomplish that the data are assessed with respect to:

- the environment
- relationships among entities
- patterns in time and space

• Main focus of this level is to infer an interpretation of the context to describe the situation in terms of indications, warnings, plans of action, etc.

• Level two is focused on the meaning of the entities detected by module at level one.

• Algorithms at this level come from automated reasoning and artificial intelligence
JDL Process Model (Level 3)

*Level three (impact assessment)* processing performs threat refinement for military or intelligence fusion systems.

- The meaning of the fused data are interpreted from an adversarial view, to infer the enemy's threat assessment.
- More in general, at this stage a *prediction* of future events/situations is fulfilled.
- For non-military applications possible risks, damages, relevant events are foreseen.
- Automated reasoning, artificial intelligence, predictive modelling and *statistical estimation* are used for these tasks.
JDL Process Model (Level 4)

**Level four (process refinement)** is a meta-process used to monitor the data fusion process to optimize data acquisition and processing.

- Data collection assets in the system are evaluated prioritizing the tasking and monitoring sensor health and status in a **feedback loop**.
- Sensor modelling, computation of **performance measures** and optimization of resources are accomplished.
- Usually these needs are conflicting and must be resolved by this stage or by a human operator.
- The modality to achieve efficient and personalized interaction with the operator are studied in this level
Level five (cognitive refinement) was proposed to stress the importance of addressing humans in the loop decision process.
JDL Process Model (Level 5)

**Level five - Cognitive Refinement (cont.)**

The output of the system are affected by how the humans perceive the output and by their intervention and control.

The following elements are to be taken into account:

- human-computer interaction
- cognitive aids to reduce human biases
- individual user preferences and constraints
- automated interpretation of data using human sensory mechanism
- search engines
- ...
JDL Process Model (Database Management)

- In a data fusion system functions to perform Database management, man-machine interface and evaluation are needed.
- A large **amount of data** is usually required by Data Fusion system to accomplish its tasks:
  - information of previous inputs, outputs, events, situations, etc.
  - a priori information
  - ...
- The importance of the database management is confirmed by the fact that **Database Management Support (DBMS)** functions may constitute as much as 25% of an implemented data fusion system
Other Functional Models

- Other models have been proposed in the literature to describe data fusion processes.
- **Dasarathy’s model** [1] categorized data fusion functions in terms of types of inputs and outputs data/information (e.g. data, features, objects).

<table>
<thead>
<tr>
<th>INPUT</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>Data</td>
</tr>
<tr>
<td>Data</td>
<td>Signal Detection</td>
</tr>
<tr>
<td>Features</td>
<td>Model-based Detection/Feature Extraction</td>
</tr>
<tr>
<td>Objects</td>
<td>Model-based Detection/Estimation</td>
</tr>
</tbody>
</table>

Other Functional Models

• The “Omnibus” process model [1] is based on the Boyd control loop that describes the human reasoning strategy.

Architectural Models
• The transformation of a functional model into a physical system requires significant efforts for the design both for what concern hardware and software. Relevant elements in system design are:
  ▪ algorithms employed
  ▪ throughput constraints
  ▪ computer resource requirements
  ▪ system flexibility
  ▪ robustness/redundancy
  ▪ …

• According to where the fusion process is accomplished two types of architecture can be distinguished:
  ▪ centralized
  ▪ distributed
Centralized Data fusion

- Data are gathered and processed together in a **centralized computing unit**.
- Intelligence is **concentrated** in one part of the system.
Distributed Data fusion

- For each sensor raw data are processed *locally*. A preliminary decision can be taken independently for each sensor data flow.
- Final decision/output is determined combined all the preliminary decisions or processed data.
Data Fusion System Architectures

Distributed vs. Centralized Data fusion

Distributed Data fusion allows:

- **Computational load** is distributed on several units
- Higher **flexibility**; the processing units can be in distributed on all the architecture according to the bandwidth requirements, spatial constraints, redundancy needs, etc.
- Higher **modularity**, scalability and reliability; the introduction and removal of devices affect less the architecture
- Less **information** transmitted on the network connecting the devices composing the system
Mathematical Models
Mathematical Models can be described considering their possible usage at different levels that are related to different type of data:

- **Signals**: measurements/observations gathered by sensors
- **Objects**: entities detected and located by processing signals
- **Situations**: interaction between entities
- **Impacts**: consequences of the projection of the current situations into the future
General Level Structure

Processing at each of Data Fusion levels involves that, in general on the available data for fusion four tasks are accomplished:

1- Data Preparation:

- It includes common formatting, spatio-temporal alignment, and confidence normalization.

- Before other fusion tasks observed data must be transformed into a suitable form to allow further processing. Generally a spatial and temporal alignment is necessary to map data into an appropriate coordinate system.
2- Data Association:

• It consists in the *generation, evaluation and selection* of association hypotheses, i.e. of hypotheses as to the applicability of specific data to particular aspects of the state estimation problem.

• Since observations can be generated by different entity, data must be grouped in a way that they can be connected to an unique entity.
3- State Estimation and Prediction:

- It consists in estimating the presence, attributes, inter-relationships, utility and performance or effectiveness of entities of interest, as appropriate to the data fusion node.

- Using physical model of the entity and statistical model of observations is possible to map observational data to a set of variables describing the “state” of the entity that is called state vector.
4- Identity Declaration:

• It consists in transforming observed attributes of an entity such as shape, size and spectral characteristics into a label that describes or names the **identity** of the entity.

• Signal data collected by sensors are processed to extract key features. These features are used by **pattern recognition** algorithms or classifiers to transform data into a declaration of identity.
The Data Fusion levels according to the JDL model [1] (1998) and the operations fulfilled in each of them are the following:

- **Sub-Object Data Assessment**: estimation and prediction of signal observable states on the basis of pixel/signal level data association and characterization.
Processing in Data Fusion Levels

- **Object Assessment**: estimation and prediction of entity states on the basis of inferences from observations.
• **Situation Assessment**: estimation and prediction of entity states on the basis of inferred relations among entities.

Processing in Data Fusion Levels

• **Impact Assessment**: estimation and prediction of effects on situations of planned or estimated/predicted actions by the participants (e.g. assessing susceptibilities and vulnerabilities to estimated/predicted threat actions given one’s own planned actions).
Processing in Data Fusion Levels

- **Process Refinement**: adaptive data acquisition and processing to support mission objectives.

Processing in Data Fusion Levels

- In each level different **processes** can be identified
- The division with respect to the **estimation process** is related to the type of entity for which the state is estimated

<table>
<thead>
<tr>
<th>Data fusion level</th>
<th>Association process</th>
<th>Estimation process</th>
<th>Entity estimated</th>
</tr>
</thead>
<tbody>
<tr>
<td>L0 - Sub-Object Assessment</td>
<td>Assignment</td>
<td>Detection</td>
<td>Signal</td>
</tr>
<tr>
<td>L1 - Object Assessment</td>
<td>Assignment</td>
<td>Attribution</td>
<td>Individual object</td>
</tr>
<tr>
<td>L2 - Situation Assessment</td>
<td>Aggregation</td>
<td>Relation</td>
<td>Aggregation (Situation)</td>
</tr>
<tr>
<td>L3 - Impact Assessment</td>
<td>Aggregation</td>
<td>Plan interaction</td>
<td>Effect (situation, given plans)</td>
</tr>
<tr>
<td>L4 - Process Refinement</td>
<td>Planning</td>
<td>(Control)</td>
<td>(Action)</td>
</tr>
</tbody>
</table>
Processing in Data Fusion Levels

Different types of data must be associated at each level of the JDL model comparing in correlation matrices hypothesis and observations.
Processing in Data Fusion Levels

Processing Examples

- **Level 0**: noise filtering (e.g. Gaussian smoothing), feature extraction (e.g. Principal Component Analysis – PCA) and image processing (e.g. morphologic filters).

- **Level 1**: position and speed estimation of moving targets (e.g. Kalman Filter). Identity estimation using pattern recognition algorithms.

- **Level 2**: event/activity detection obtained integrating information on single objects obtained by Level 1 processing (e.g. Neural Networks).
Processing in Data Fusion Levels

Processing Examples (cont.)

- **Level 3**: potential enemy threats, vulnerabilities, possible damages evaluation (e.g. Game Theory, Knowledge based approaches)
- **Level 4**: performance evaluation and automatic parameters adaptation (e.g. sensor, algorithm modeling).
- **Level 5**: focus of attention managing, human/machine interaction managing (e.g. expert systems, genetic algorithms, reinforcement learning)
Mathematical Models

Data Preparation
Data Alignment

• The **Data Alignment** is a fundamental process to accomplish the Data Preparation

• In a Data Fusion system the data gathered from multiple sources must be transformed into a common reference frame to allow the further association and estimation/prediction.

• Data alignment is devoted to convert acquired data (coming from heterogeneous not synchronized sources) into a common format suitable for comparing and/or combining them of data

• Data alignment at Level 1 of the JDL model operates on raw data gathered from sensors.
Data Alignment

• Data alignment algorithms include *spatial* reference adjustment, *temporal* reference adjustment, and *unit* adjustment.
• Data alignment can be a very *complex* and time-consuming process particularly if sensors vary in their typology and their number is high.
• In the following two Data Alignment issues will be described:
  ▪ Temporal alignment, e.g. **Network Time Protocol** (NTP, defined in RFC 1305)
  ▪ Spatial alignment in computer vision problems, e.g. **camera calibration**
Data Alignment - NTP

• Temporal alignment performed using protocols that provide clock synchronization through noisy channels, like **Network Time Protocol** (NTP, defined in RFC 1305)

• The NTP works as follows:
  - the client sends a timestamp (T1C) to the server
  - the server (reference clock) sends back a packet containing the time was sent (T2S), the time the first packet was received (T1S) and the original timestamp (T1C)
  - the client now has 4 timestamps
    - round trip delay = (T1S - T1C) - (T2S - T2C)
    - clock offset = ((T1S - T1C) + (T2S - T2C))/2

Data Alignment - NTP

• NTP is a so called **Cristian algorithm** [1]: there is a clock reference (GPS receiver, radio clock, atomic clock, ...) which maintains a server synchronized and all other PCs synchronized with the server.

• **Berkeley algorithms** don’t require a clock reference: the time server will periodically fetch the time from all the time clients, average the results, and then report back to the clients the adjustment that needs be made to their local clocks to synchronize the whole network.

• **Clock Sampling Mutual Network Synchronization (CS-MNS)**, are completely decentralized algorithms, more suitable for sparse and hoc networks: all clocks exchange timestamps to achieve a global synchronization.

Data Alignment – Camera Calibration

• Computer Vision addresses to recover world information (3D) from images (2D)

• One of the basic techniques in this domain is the **camera calibration** that consists in establishing a relationship between the image coordinates and the 3D world coordinates

• This process is useful to perform several 3D visual tasks as stereoscopy or motion from image. Moreover it allows to transform data acquired by multiple camera into a **common reference system**

• Several approaches are present in the literature (see [1]) most of them based on camera projection and perspective projection models.

Tsai’s Camera Model

- Tsai's camera model is based on the **pin hole model** of perspective projection.
Tsai’s Camera Model

Tsai’s model has 11 parameters to be found:

- **five internal** (also called intrinsic or interior) parameters
  - $f$ - effective focal length of the pin hole camera,
  - $k$ - 1st order radial lens distortion coefficient
  - $C_x$, $C_y$ - coordinates of center of radial lens distortion and the piercing point of the camera coordinate frame's Z axis with the camera’s sensor plane,
  - $S_x$ - scale factor to account for any uncertainty due to frame grabber horizontal scanline resampling,

- **six external** (also called extrinsic or exterior) parameters,
  - $R_x$, $R_y$, $R_z$ - rotation angles for the transform between the world and camera coordinate frames, and
  - $T_x$, $T_y$, $T_z$ - translational components for the transform between the world and camera coordinate frames.
Data Alignment - Camera Calibration

• The transformation from world \((X_w, Y_w, Z_w)\) to image \((X_i, Y_i, Z_i)\) co-ordinates considers the **extrinsic parameters** of the camera (Translation \(T\) and Rotation \(R\)) within the equation:

\[
\begin{pmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{pmatrix} = \begin{pmatrix} r_1 & r_2 & r_3 & 0 \\ r_4 & r_5 & r_6 & 0 \\ r_7 & r_8 & r_9 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} + \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix}
\]

where \(r_i\) is a function of the **Euler rotation angles** \((R_x, R_y, R_z)\) around the three axes and and \((T_x, T_y, T_z)\) are the **3D translation** parameters from world to image co-ordinates.
Data Alignment - Camera Calibration

Tsai’s Camera Model

- The transformation from **3D position** (in the image co-ordinate frame) to the **image plane** is then computed through the following steps:

  - Transformation from 3D world co-ordinates \((X_i, Y_i)\) to undistorted image plane \((X_u, Y_u)\) co-ordinates
  
  \[
  X_u = X_d \left(1 + kr^2\right) \\
  Y_u = Y_d \left(1 + kr^2\right)
  \]

  - Transformation from undistorted \((X_u, Y_u)\) to distorted \((X_d, Y_d)\) image co-ordinates

  \[
  r = \sqrt{X_d^2 + Y_d^2}
  \]

  - Transformation from distorted co-ordinates in image plane \((X_d, Y_d)\) to the final image co-ordinates \((X_f, Y_f)\) are:

  \[
  X_f = \frac{S_x X_d}{d_x} + C_x \\
  Y_f = \frac{Y_d}{d_y} + C_y
  \]

  \(d_x\) and \(d_y\) are fixed parameters of the camera depending on CCD dimension and image resolution.
Tsai’s Camera Calibration

- Camera calibration consists in the estimation of the model parameters for an un-calibrated camera. **Tsai’s algorithm** needs corresponding 3D \((X_w, Y_w, Z_w)\) point co-ordinates and 2D \((X_i, Y_i)\) pixels in the image.
- Tsai’s algorithm **n>8 points** correspondences and it solves the calibration problem with a set of n linear equations based on the radial alignment constraint.
Data Alignment - Camera Calibration

• Internal camera parameters are usually available provided by the manufacturer or computed using a known target (usually a chessboard plane), i.e. the camera is calibrated. Therefore, it will be sufficient to find six parameters relative to the position and orientation. This problem is also referred to as pose estimation.

• Two forms of calibration are possible:
  ▪ coplanar - the calibration points lie in a single plane in 3D, and
  ▪ non-coplanar - the calibration points occupy a 3D volume.
Mathematical Models

Data Association
Data Association

- The association step can be applied at every level of the JDL model to collect the output of the previous level into a common and unique representation.
- In tracking applications, data association processes are mainly employed in the level 1 (“Object Refinement”).
- When measurements originating from the same real world object are gathered to generate single tracks, a measurement-to-track association is performed. This procedure is common in centralized data fusion architecture where multiple sensors send their measurements to a central
Example: measurement-to-track association

- **Measurements** are the outputs of a change detection module applied on frames collected by cameras
Data Association

- In the case of distributed systems it is also possible to compare tracks from different processing nodes estimating the state of the same real world object. This is refered to as track-to-track association.
Data Association

• A large variety of approaches can be found to deal with these problems. For example Nearest Neighbour, Probabilistic Data Association Filter (PDAF), Joint Probabilistic Data Association Filter (JPDAF), Lagrangian Relaxtion and Neural Networks can be used for this task [1].

• Data association and state estimation are usually coupled in tracking frameworks. Typical examples are the PDAF and JPDAF that integrates a data association strategy in the Kalman Filter algorithm.

Data Association

Data Association can be divided into three subfunctions:

- **Hypothesis Generation**: available data are used to generate association hypothesis (tracks) via feasibility gating of prior hypothesis or via data clustering.

- **Hypothesis Evaluation**: these hypotheses are evaluated for self consistency using kinematic, parametric, attribute, ID, and a priori data.

- **Hypothesis Selection**: a search is performed to find the preferred hypotheses based either on the individual track association scores or on global scores.
Hypothesis Generation

- In this phase the search space to be considered for the association is reduced by generating hypothesis according to defined models.

- For target tracking typically spatio-temporal relational models are employed to gate, prune, aggregate data. For example kinematics model can be used for predicting area of investigation.

- This task is usually suboptimal and then it should be conservative, admitting more false alarm rather than eliminating possible true ones.
Data Association

Hypothesis Evaluation

• This process aims at assigning a **score** to hypothesis resulting from the previous process (hypothesis generation).

• Hypothesis evaluation techniques are related to the accuracy of the uncertainty modeling. To this end several concepts can be used, for example **Bayesian probability, fuzzy membership**, information theory, etc.

• Hypothesis evaluation is sometimes combined with **state estimation** where the uncertainty on the state is used as evaluation score. The likelihood ratio is particularly suited for this.
Data Association

Hypothesis Selection

• Hypothesis selection involves searching the scored hypotheses to **select** one or more to be used for state estimation.

• Hypothesis selection eliminates, split, combines or confirms association hypotheses to maintain or delete tracks.

• Typically hypothesis generation techniques provide an expectation of the observed phenomenon, hypothesis evaluation ones **assess** it with respect to observation and hypothesis selection methods establish reliable hypothesis to state estimation module.
Nearest Neighbor

- **Nearest Neighbor** is the simplest form of the association algorithm

- In tracking applications, the measurement that is nearest to the track is chosen to perform the updating process, i.e. to confirm the position prediction.

- This algorithm is simple and with low **computational cost** but it tends to fail in dense environments
Data Association - Algorithms

Probabilistic Data Association Filter - PDAF

• The **PDAF** is an algorithm developed by Bar-Shalom and Tse [1], that jointly performs data association and state estimation for single target tracking.

• Unlike Nearest Neighbor data association, all measurements in a **gated region** around a track are considered as possible updates for the target.

• Proper **target-measurement** correspondences are maintained by continually computing the association probabilities of the various possible associations. This process is integrated in the **Kalman Filter** algorithm.

Data Association - Algorithms

Probabilistic Data Association Filter - PDAF

• Kalman Filter is an efficient method for tracking via state estimation when the distribution on measurements is Gaussian. This is not verified if measurement noise is generated by multiple observations or if the measurement disappear.

• PDAF is an extension of the Kalman Filter that uses a Bayesian approach to data association to update the state when there is a single target and possibly no measurements or multiple observations.

• Rather than possibly erring by choosing the data closest to what is expected (nearest neighbor) to update the state the influence of the various candidate measurements is used as a weight factor.
Joint Probabilistic Data Association Filter - JPDAF

• PDAF does not consider situations where noise is caused by other targets. This assumption makes it **fails in multitarget cluttered situations**

• JPDAF extends the PDAF to the multitarget case by computing the measurement-target association jointly across all the targets.

• The joint computation of the target-measurement association **prevents more trackers to be latch onto the same target**

• The key notion of JPDAF is that of a **joint event** whose probability depends on the target predicted measurement and the actual measurement which it associated with.
Graph-based Data Association

• **Target position** for each frame can be represented as a **node** of a bipartite oriented graph. Partitions of the graph are the targets detected in each frame.

• **Edges** are defined as a **similarity measure** between targets between successive frames.

Node = target

Edge = similarity measure

Targets at frame k
Graph-based Data Association

- A **track** is constituted by a **path**, that is set of node, not necessarily belonging to adjacent partitions, connected by edges whose weight is a similarity measure between targets observed in different instants.

- The computation of the best set of tracks can be obtained using the **maximum weight path cover**, i.e. finding the combination of paths entailing the maximum total weight over all the edges.

- Several **optimization algorithms** (e.g. [1]) are present in the literature to compute the maximum weight path cover considering the possibility of false and miss detection.

Data Association - Algorithms

Graph-based Data Association - Example

• The thickness of the edges represent the weight value compute between targets detected in different frames

• The association allows one to overcome a miss detection occurred at frame 2
Mathematical Models

State Estimation
State Estimation

• The estimation problem attempts to find the value of a state vector that best fits, in a mathematical sense, the observations.

• The state vector contains the relevant information required to describe the system under investigation. For example in tracking problems it can be related to kinematic (position, velocity) characteristics.

• The estimation problem can be dynamic if the vector state changes as a function of the time, or static when it is constant in time.

• Here, the attention will be focus on dynamic state estimation since this is generally the situation encounter for tracking applications.
• Bayesian Filtering is a successful approach to the problem of sequential estimation of the state of a dynamic system by using a sequence of noisy measurement.

• In order to analyze and make inferences about a dynamic system usually two models are required:

  ▪ system or dynamic model: it describes the evolution of the state with time

    \[ x_k = f_{k-1}(x_{k-1}, n_{k-1}) \]  
    \[ x_k \in \mathbb{R}^{n_x} \]  

  ▪ measurement model: it relates the noisy measurements with to the state

    \[ z_k = h_k(x_k, w_k) \]  
    \[ z_k \in \mathbb{R}^{n_z} \]
• Assuming that the models to be described in a **probabilistic form**, the Bayesian approach provides a rigorous general framework for dynamic state estimation

• The main idea of Bayesian approach to dynamic state estimation is to construct the **posterior** probability density function (pdf) of the state $x_k$ based on all available information, including the sequence of received measurements $Z_k = z_{1:k}$

\[
p(x_k | Z_k) \rightarrow Z_k = z_1, \ldots, z_k
\]

• In principle the **optimal estimate** of the state can be obtained from the posterior pdf
State Estimation – Bayesian Filtering

- **Recursive filtering** provides an estimate sequentially every time that a new measurement is available. This solution is well suited for the tracking problem where state estimate (position and velocity) is usually required for each time interval.

- Two basic stages are involved in the filtering process:
  - **prediction**: the system model is used to predict the state pdf from a time instant to the subsequent
    \[
    p(x_k \mid Z_{k-1}) = \int p(x_k \mid x_{k-1}) p(x_{k-1} \mid Z_{k-1}) dx_{k-1}
    \]
  - **update**: the last measurement is used to modify (correct) the predicted pdf which may be not accurate due to the noise affecting the state
    \[
    p(x_k \mid Z_k) = p(x_k \mid z_k, Z_{k-1}) = \frac{p(z_k \mid x_k, Z_{k-1}) p(x_k \mid Z_{k-1})}{p(z_k \mid Z_{k-1})} = \frac{p(z_k \mid x_k) p(x_k \mid Z_{k-1})}{p(z_k \mid Z_{k-1})}
    \]
State Estimation – Bayesian Filtering

- The knowledge of the posterior pdf allows one to compute optimal state estimates with respect to any criterion. For example the MMSE (Minimum Mean Square Error) and MAP (Maximum A Posteriori) can be computed as:

\[
\hat{x}_{k|k}^{\text{MMSE}} = \mathbb{E}\{x_k \mid Z_k\} = \int x_k \cdot p(x_k \mid Z_k) \, dx_k
\]

- In general cases the solution of the filtering process can’t be computed analytically. In the following the situation in which an analytical solution is available are describe together with the suboptimal solutions for the other cases.
State Estimation – Kalman Filter

- Optimal finite-dimensional algorithm for recursive Bayesian state estimation are available in linear-Gaussian cases, provided by the **Kalman Filter**, and in few other more specific and not common situations.

- The Kalman filter assumes that the posterior pdf at every step is **Gaussian** and then it can be completely characterized by the **mean** and the **covariance**.

- These assumptions hold if $n_{k-1}$ and $w_k$ are drawn from **Gaussian** density and the functions $f_{k-1}$ and $h_k$ are **linear**. Consequently (1) and (2) become

\[
\begin{align*}
x_k &= F_{k-1} x_{k-1} + n_{k-1} \\
z_k &= H_k x_k + w_k
\end{align*}
\]
State Estimation – Kalman Filter

- The matrices $F_{k-1}$ (of dimension $n_n \times n_n$) and $H_k$ (of dimension $n_z \times n_x$) define the **linear** functions. Random sequences $n_{k-1}$ and $w_k$ are mutually independent zero-mean white Gaussian with covariance $Q_{k-1}$ and $R_k$ respectively.

- **Mean and covariance** predictions are computed respectively as:

$$
\hat{x}_{k|k-1} = F_{k-1} \hat{x}_{k-1|k-1}
$$

$$
P_{k|k-1} = Q_{k-1} + F_{k-1} P_{k-1|k-1} F_{k-1}^T
$$

- In the update process the prediction is compared to the observation. The resulting term $v_k$ is called **innovation** and its covariance $S_{k-1}$ can be computed as:

$$
v_k = z_k - H_k \hat{x}_{k|k-1}
$$

$$
S_k = H_k P_{k|k-1} H_k^T + R_k
$$
State Estimation – Kalman Filter

• The Kalman gain is computed according to the predicted covariance and the innovation covariance as:

\[ K_k = P_{k|k-1} H_k^T S_k^{-1} \]  \hspace{1cm} (5)

• The resulting estimated mean and covariance results equal to:

\[ \hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k (z_k - H_k \hat{x}_{k|k-1}) \]  \hspace{1cm} (6)

\[ P_{k|k} = P_{k|k-1} - K_k S_k K_k^T = [I - K_k H_k] P_{k|k-1} \]  \hspace{1cm} (7)

• When the Kalman gain has an high value, i.e. when the \( P_{k|k-1} \) is large or the \( S_k \) is small, the innovation is considered reliable.

• A large Kalman gain value occurs when the prediction is not consistent and/or the innovation is trustworthy and it implies that the estimate \( \hat{x}_{k|k} \) relies more on the innovation than on prediction
Example
Observations (state+Gaussian noise): $\mathbf{z}_k = [x; y]^T \in \mathbb{R}^2$

$\mathbf{z}_1 = [0.1 ; 0.05]^T$
$\mathbf{z}_2 = [1.09 ; 1.04]^T$
$\mathbf{z}_3 = [2.1 ; 2.11]^T$
$\mathbf{z}_4 = [3.1 ; 2.1]^T$
$\mathbf{z}_5 = [4.02 ; 2.05]^T$
$\mathbf{z}_6 = [5.04 ; 2.02]^T$
State Estimation – Kalman Filter

Example cont’d

State: kinematics characteristics
\( x_k = [x; y; v_x; v_y]^T \in \mathbb{R}^4 \)

Transition Equation
\[
\begin{bmatrix}
1 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
x_k \\
y_k \\
v_{x,k} \\
v_{y,k} \\
\end{bmatrix}
= \begin{bmatrix}
n_x \\
n_y \\
n_{v_x} \\
n_{v_y} \\
\end{bmatrix}
\]

Noise matrices: defined according to a-priori assumptions on state and observation noise
\[
Q_k = \begin{bmatrix}
0.1 & 0 & 0 & 0 \\
0 & 0.1 & 0 & 0 \\
0 & 0 & 0.1 & 0 \\
0 & 0 & 0 & 0.1 \\
\end{bmatrix}
\]
\[
R_k = \begin{bmatrix}
0.1 & 0 \\
0 & 0.1 \\
\end{bmatrix}
\]

Initial State: \([0; 0; 1; 1]^T\)

Observation Equation
\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
\end{bmatrix}
\begin{bmatrix}
x_k \\
y_k \\
\end{bmatrix}
= \begin{bmatrix}
wx \\
w_y \\
\end{bmatrix}
\]
State Estimation – Kalman Filter

Example cont’d

\[ x_1 = [0 \ ; \ 0 \ ; \ 1 \ ; \ 1]^T \]
\[ x_2 = [1.05 \ ; \ 1.02 \ ; \ 1 \ ; \ 1]^T \]
\[ x_3 = [2.08 \ ; \ 2.08 \ ; \ 1.02 \ ; \ 1.03]^T \]
\[ x_4 = [3.1 \ ; \ 2.3 \ ; \ 1.02 \ ; \ 0.62]^T \]
\[ x_5 = [4.04 \ ; \ 2.21 \ ; \ 0.98 \ ; \ 0.26]^T \]
\[ x_6 = [5.04 \ ; \ 2.1 \ ; \ 0.99 \ ; \ 0.07]^T \]
The **Extended Kalman Filter (EKF)** is a suboptimal solution of the Bayesian filtering problem for nonlinear systems with additive noise.

\[
\begin{align*}
x_k &= f_{k-1}(x_{k-1}) + n_{k-1} \\
z_k &= h_k(x_k) + w_k
\end{align*}
\]

Random sequences \(n_{k-1}\) and \(w_k\) are mutually independent zero-mean white **Gaussian** with covariance \(Q_{k-1}\) and \(R_k\) respectively.

The basic idea is to **linearize the non-linear functions** by the first term in the Taylor series expansion.

**Prediction** and **innovation** are computed using, respectively, non-linear functions \(f_{k-1}\) and \(h_k\).
• The other formulas for performing the estimation coincides with the ones of the **Kalman Filter** (5)-(7) substituting $F_{k-1}$ and $H_k$ with their local linearization:

$$
\hat{F}_k = \left[ \nabla_{x_{k-1}} f^T_k (\hat{x}_{k-1}) \right]^T_{x_{k-1}=\hat{x}_{k-1|k-1}}
$$

$$
\hat{H}_k = \left[ \nabla_{x_k} h^T_k (\hat{x}_k) \right]^T_{x_k=\hat{x}_{k|k-1}}
$$

where

$$
\nabla_{x_k} = \begin{bmatrix} \frac{d}{dx_k[1]} & \cdots & \frac{d}{dx_k[n_x]} \end{bmatrix}^T
$$

with $x_k[i]$ is the $i$-th component of the state vector.
State Estimation – UKF

• The **Unscented Kalman Filter (UKF)** handles the same non-linear Gaussian problem as EKF but without the linearization procedure. Instead UKF approximates the posterior \( p(x_k|Z_k) \) by a Gaussian density which is represented by a **set of deterministically chosen sample points**.

• UKF is based on the idea of **statistical linearization** rather than analytical linearization as for EKF. Statistical linearization is performed via the linear regression through (sample) points, i.e. the computation of a model using the available data.

• Different approaches are present to **select the sample points** in terms of their number, value and weight.
State Estimation – Particle Filter

• The **Particle Filter (PF)** aims at computing an approximated solution of the Bayesian filtering problem in **non-Gaussian non-linear** cases.

• Worth of noticing is that KF, EKF, UKF, etc. are **normally used** in non-Gaussian non-linear applications but, in these cases, they don’t provide the optimal solution for KF, or the best solution for EKF and UKF.

• Particle filter approximates the filtered posterior distribution by a **set of weighted samples (particles)**. Particles are **propagated** over time by a transition model and they are weighted according to a **likelihood score** that takes into account the observations.
State Estimation – Particle Filter

- A **Sequential Monte Carlo (SMC) estimation** is performed on the particles that represent the pdf using the sequential importance sampling approach.

- The **Monte Carlo integration** method is the basis of the SMC approaches. MC integration consists in factorizing the function $g(x)$ to be integrated as $g(x) = f(x) \pi(x)$ where $\pi(x)$ is a pdf. Then **samples are drawn from** $\pi(x)$ and the integral is approximated as a sum of weighted samples.

- **Sequential Importance Sampling (SIS) algorithm** applies a MC integration method called **importance sampling** to perform non-linear filtering.
State Estimation – Particle Filter

• The principle of the **importance sampling (IS)** consists in drawing a set of \( N \) candidate samples (i.e. particles) \( \{ x_k^i \}_{i=1}^N \) from a so called **proposal distribution** (or importance distribution) \( q(x_{0:k-1}|z_{1:k}) \) instead of directly from the posterior.

• Importance distribution is **easier to be modeled** than posterior and allows efficient sampling procedures.

• A **correcting (weighting or updating)** procedure is then applied to take into account the difference between the proposal and the posterior, according to:

\[
W_k^i \propto \frac{p(x_{0:k-1}^i|z_{1:k})}{q(x_{0:k-1}^i|z_{1:k})}
\]
State Estimation – Particle Filter

- Due to the recursive nature of the problem, assuming a first order Markovian dependency of the posterior with respect to the state and the independence between observation at subsequent time instant, the samples can be drawn from an importance distribution:

\[ q(x^i_k \mid x^i_{k-1}, z_k) \]

- Under these assumptions the weight is computed as:

\[ w^i_k = \frac{p(z_k \mid x^i_k) p(x^i_k \mid x^i_{k-1})}{q(x^i_k \mid z_k, x^i_{k-1})} \]
State Estimation – Particle Filter

- The **posterior is approximated** as:

\[
p(x_k | z_k) \approx \sum_{i=1}^{N} w_k^i \delta(x_k - x_k^i)
\]

- Weighted samples \( \approx p(x_k | z_k) \)

- Particles sampled from \( q(x_k^i | x_{k-1}^i, z_k) \)
State Estimation – SIS Particle Filter

- **Sequential Importance Sampling (SIS)** scheme is a widely used algorithm based on these assumptions. The **pseudo-code** of SIS particle filter is here described:

\[
\left\{ x_k^i, w_k^i \right\}_{i=1}^N = \text{SIS} \left[ \left\{ x_{k-1}^i, w_{k-1}^i \right\}_{i=1}^N, z_k \right]
\]

FOR \( i = 1: N \)
- Draw \( x_k^i \) \( \sim q(x_k^i \mid x_{k-1}^i, z_k) \)
- Evaluate the importance weights up to a normalizing constant as

\[
w_k^i = \frac{p(z_k \mid x_k^i) p(x_k^i \mid x_{k-1}^i)}{q(x_k^i \mid z_k, x_{k-1}^i)}
\]

END FOR

- Calculate total weight: \( t = \text{SUM} \left[ \left\{ \tilde{w}_k^i \right\}_{i=1}^N \right] \)
- FOR \( i = 1: N \)
  - Normalize: \( w_k^i = t^{-1} \tilde{w}_k^i \)
- END FOR
State Estimation – Particle Filter

Degeneracy problem - Resampling

- The **accuracy** of the particle filter approximation is related to the similarity of the importance sampling function with the posterior distribution.

- The assumptions on the importance sampling function made in the SIS particle filter lead to **degeneracy problems** consisting in the increase of the variance of the importance weights.

- After a certain number of recursive steps all but one particle will have **negligible weights**

- To avoid this problem **resampling** procedures are proposed in order to redistribute particles in order to **eliminates** samples with low weights and **multiplies** those with high importance weight $w_k^i$. 
State Estimation – Particle Filter

• Several approaches exist in the literature to accomplish the **resampling step**. However in general the idea is to redistribute the particles in a proper way giving to each of them the same weight:

\[
\left\{ \tilde{X}^i_k, \frac{1}{N} \right\}^N_{i=1} \sim \left\{ X^i_k, w^i_k \right\}^N_{i=1}
\]

\[
\left\{ X^i_k, w^i_k \right\}^N_{i=1}
\]

\[
\left\{ \tilde{X}^i_k, \frac{1}{N} \right\}^N_{i=1}
\]

State Estimation – SIR Particle Filter

- The sampling procedure on general importance sampling functions $q(x_k^i|x_{k-1}^i,z_k)$ can be computationally onerous for real-time applications (e.g. tracking). Then several approaches have been proposed to have efficient and accurate particle filter schemes.

- In the Sequential Importance Resampling (SIR) filter (also known as bootstrap filter) samples are drawn from the a priori distribution $p(x_k|x_{k-1}^i)$ (then it is assumed to be the importance sampling function).

- In tracking application $p(x_k|x_{k-1}^i)$ can be defined as the motion model of the target that is based on the assumptions that can be made on the target movements.
State Estimation – SIR Particle Filter

**SIR Particle Filter pseudo-code**

\[
\left[ \{ x^i_k \}_{i=1}^N \right] = \text{SIR} \left[ \{ x^N_{k-1} \}_{i=1}^N , z_k \right]
\]

- FOR \( i = 1 : N \)
  - Draw \( x^i_k \) \( \sim \) \( p(x_k | x^i_{k-1}) \)
  - Calculate \( \tilde{w}^i_k = p(z_k | x^i_k) \)
- END FOR

- Calculate total weight: \( t = \text{SUM} \left[ \{ \tilde{w}^i_k \}_{i=1}^N \right] \)
- FOR \( i = 1 : N \)
  - Normalize: \( w^i_k = t^{-1} \tilde{w}^i_k \)
- END FOR

- Resample: \( \left[ \{ x^i_k, -,- \}_{i=1}^N \right] = \text{RESAMPLE} \left[ \{ x^i_k, w^i_k \}_{i=1}^N \right] \)
State Estimation – SIR Particle Filter

Graphical Example

\[
\left\{ \tilde{x}_i^k, \frac{1}{N} \right\}_{i=1}^N \quad \left\{ x_{k+1}^i, \frac{1}{N} \right\}_{i=1}^N
\]

\[
\left\{ \tilde{x}_i^k, \frac{1}{N} \right\}_{i=1}^N \quad \left\{ x_{k+1}^i, w_k^i \right\}_{i=1}^N
\]
Multiple Hypothesis

- In several works the Particle Filter approaches are referred to as *multiple hypothesis* methods.

- Each particle can be interpreted as an *hypothesis of the state* value and the correspondent way is its probability of being the right one.

- For example in a *tracking application* each particle represents a possible position of the object and its weight is the probability that the object is in that place.
State Estimation – Particle Filter

Multiple Hypothesis: Example

\[ x_{k|k-1} \]

Probability hypothesis 1

Probability hypothesis 2

Probability hypothesis \( N_s \)

Estimated Posterior
State Estimation – PDAF

- **Probabilistic Data Association Filter - PDAF** (already described in the Data Association section) integrates in the **Kalman Filter** framework a data association technique in order to appropriately perform the update step when more measurements are present.

- The prediction of the mean and the covariance and the innovation covariance is performed as in the **standard filter**:

\[
\hat{x}_{k|k-1} = F_{k-1} \hat{x}_{k-1|k-1}
\]

Predicted mean

\[
P_{k|k-1} = Q_{k-1} + F_{k-1} P_{k-1|k-1} F_{k-1}^T
\]

Covariance of the predicted state

\[
S_k = H_k P_{k|k-1} H_k^T + R_k
\]

Innovation covariance
State Estimation – PDAF

- At each time a **validation region** is established to select the observation to consider.

  \[ \hat{z}_{k|k-1} = H_k \hat{x}_{k|k-1} \]

  \[ V_k = \{ \mathbf{Z} : [\mathbf{z} - \hat{\mathbf{z}}_{k|k-1}]^T S_k^{-1} [\mathbf{z} - \hat{\mathbf{z}}_{k|k-1}] \leq \gamma \} \]

- Among the validated measurements one is considered as to be generated by the target and **all the other are assumed to be clutter** spatially independent identically distributed (i.i.d.).

- The PDAF uses a decomposition of the estimation with respect to the measurements of the validated region \( \mathbf{Z}_k = \{ \mathbf{z}^i_k \} \)
  \( i = 1, \ldots, m_k \). The estimation is performed **conditioning it to the validated measurements**.
State Estimation – PDAF

• The **estimate conditioned on measurement** *i* being correct is:

\[
\hat{x}_{k|k}^i = \hat{x}_{k|k-1} + K_k v_k^i, \quad i = 1, \ldots, m_k
\]

\[
v_k^i = Z_k^i - \hat{Z}_{k|k-1}^i
\]

Innovation

where the **Kalman gain** \(K_k\) is the same as in the standard formulation (Eq. 5).

• The **state update equation** of the PDAF computed over all the validated measurements is

\[
\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k v_k
\]

\[
v_k = \sum_{i=1}^{m_k} \beta_k^i v_k^i, \quad \beta_k^i \equiv P\{\theta_k^i | Z_k\}
\]

where \(\theta_k^i\) is the **association event**, i.e. it considers that \(z_k^i\) is the target originated measurement.
State Estimation – PDAF

• The term $\beta_k^i$ is derived from the number of measurements in the validation region and from their location

• The **covariance** associated with the update state is:

$$ P_{k|k} = \beta_k^0 P_{k|k-1} + \left[1 - \beta_k^0\right] P_{k|k}^c + \tilde{P}_k $$

where $\beta_k^0$ is related to the event of none measurements originated by the target, and

$$ P_{k|k}^c = P_{k|k-1} - K_k S_k K_k^T $$

is the covariance of the state updated with the **correct measurement**.

• The **spread of the innovations term** is given by:

$$ \tilde{P}_k = K_k \left[ \sum_{i=1}^{m_k} \beta_k^i v_k^i v_k^i T - v_k v_k^T \right] K_k^T $$
Joint Probabilistic Data Association Filter - JPDAF (also introduced in the Data Association section) provides an optimal solution to the data association problem.

- It is an extension to the PDA filter which adds a logic layer to ensure that tracks are correctly distributed over the measurements.

- There are two assumptions done by the JPDAF filter:
  - The measurement process generates, at most, one observation for each peak in the likelihood.
  - And each target induces, at most, one peak in the likelihood.
State Estimation – JPDAF

• The main idea is to jointly calculate the update step of multiple trackers in order to exclude unfeasible assignments like:
  • One target generating two measurements
  • Two targets generating the same measurement
• One drawback of **JPDAF** is that the number of possible hypothesis associating different observations with targets increases exponentially with the number of tracked object and clutter.
State Estimation – JPDAF

• All equations in **JPDAF** are the same to the equations of **PDAF** except to one.

• In the calculation of $\beta_{k}^{i}$, **PDAF** assumes that all tracks are isolated, whereas in **JPDA**, the probability of track “j” being associated with measurement “i” is the sum of the probabilities of all joint events $\theta_{k}$ in which track “j” is associated with observation “i”. It can be express as:

$$
\beta_{k}^{i,j} = \sum_{\theta_{k}} P\{\theta_{k}^{i,j} | Z_{k}\} \ast w_{i,j}^{j} \{\theta_{k}\}
$$

• Where $w_{i,j}^{j}$ is a binary variable indicating whether track “j” is associated with measurement “i” in the event $\theta_{k}$
Multitarget Bayesian Filtering

• Most of the state estimation methods described before are designed to handle **single target state**, i.e. to estimate the behaviour of a target independently to the surrounding environment.

• The Multitarget Bayes Filter aims at jointly estimating the **state of multiple targets**. The multitarget state is composed by the integration of the single target states.

• The Multitarget Bayes filter propagates the **multitarget posterior density** to find the optimal solution (in terms of minimum covariance error) of the filtering problem.

• The computational cost of these methods grows **exponentially** with the number of targets.
Random Finite Sets

• A major purpose of Finite Set Statistic (FISST) is to extend the idea to propagate the expectation of the posterior density to multi-target problem.

• The problem they have to handle is that Multi-target state and measurement have unknown dimension, changing in time, at any time new targets may appear and existing targets may disappear.

• The Multi-target state and measurement are the collection of the state and the measurement of each target.

\[
X_k = [x_1^k, \ldots, x_{M(k)}^k] \\
Z_k = [z_1^k, \ldots, z_{N(k)}^k]
\]

Variable for each time instant \(k\)
Random Finite Sets

- Uncertainties in the multi-target state $X$ and observation $Z$ are described respectively by the Random Finite Sets (RFSs) $\Xi$ and $\Sigma$

- The RFS $\Xi$ encapsulates all aspects of multi-target motion such as the time-varying number of targets, individual target motion, target birth, spawning and targets interactions

- The RFS $\Sigma$ encapsulates all sensor characteristics such as measurements noise, sensor field of view (i.e. state-dependent probability of detection) and false alarms.
Random Finite Sets

- The general procedure for the Multi-target Bayesian Filtering is described in the following scheme:
The approach previously described can be extended to a **multisource scenario**. Suppose there are $s$ sensors in the scene. The **states of the sensors** are denoted by:

\[ X = \left\{ *_{1}, \ldots, *_{s} \right\} \]

The **total observation set** collected by all sensors has the form:

\[ Z = \left\{ Z^{1} U \ldots U Z^{s} \right\} \]

where $U$ is the **disjoint union** operator.
Multisource Measurement Model

• Assuming that the measurements are conditionally independent of target states, the likelihood function is:

\[ f_k(Z | X, X) = f_k(Z | X, x) \cdots f_k(Z | X, x) \]

\[ = f_k(Z | X) \cdots f_k(Z | X) \]

where \( f_k(Z | X, x) \) is the multitarget likelihood for the \( j \)-th sensor

• Simultaneously arriving multisource data can thus be processed in the usual manner using the multitarget corrector (updating) equations.
Bayes Filter

• The prediction and updating steps of the Multitarget Bayes filtering are given by:

\[ p_{k|k-1}(X_k \mid Z_{1:k-1}) = \int f_{k\setminus k-1}(X_k \mid X) p_{k-1|k-1}(X \mid Z_{1:k-1}) \, dx \]

Prediction Step

\[ p_k(X_k \mid Z_{1:k}) = \frac{g_k(Z_k \mid X_k) p_{k|k-1}(X \mid Z_{1:k-1})}{\int g_k(Z_k \mid X) p_{k|k-1}(X \mid Z_{1:k-1}) \, dx} \]

Update Step

• The recursion involves the evaluation of multiple integrals on the state space

• The size of the state-space grows \textit{exponentially} with the number of targets
Bayes Filter propagates the **posterior probability**. This in general cases is not feasible.

The **Probability Hypothesis Density (PHD)** Filter propagates only the **first order moment of the multi-target density**, named PHD.

The PHD is the density $D_{k\mid k}(X \mid Z^{(k)})$ whose integral

$$\int_S D_{k\mid k}(x \mid Z^{(k)}) \, dx$$

on any region $S$ of the state space gives the **expected number of targets** contained in $S$. 

**Relation Bayes Filter – PHD Filter**
Relation Bayes Filter – PHD Filter

- The PHD function is defined over the state space of one target.
- It requires the computation of integrals that have not a closed form in general.
- The PHD Filter has not data-association functionality.

\[
p_{k|k}(X|Z^k) \rightarrow p_{k+1|k}(X|Z^k) \rightarrow p_{k+1|k+1}(X|Z^{k+1}) \rightarrow \ldots
\]

\[
D_{k|k} \rightarrow D_{k+1|k} \rightarrow D_{k+1|k+1} \rightarrow \ldots
\]

\[Z_{k+1}\]
We approximate the PHD function with a set of particles, using Particle Filter.
Particle PHD filter: Prediction Step

- The transition function is given by:

  \[ \phi_{k|k-1}(x, \xi) = e^\mathbf{A} f_{k|k-1}(x | \xi) \]

  Probability that the target still exists at time \( k \)

- Sample

  \[ \tilde{x}^{(i)}_k \approx \begin{cases} q_k(\cdot | \tilde{x}^{(i)}_{k-1}, Z_k), & i = 1, \ldots, L_{k-1} \\ p_k(\cdot | Z_k), & i = L_{k-1} + 1, \ldots, L_{k-1} + J_k \end{cases} \]

  New born targets

- And compute the predicted weight

  \[ \tilde{w}_{k|k-1}^{(i)} = \begin{cases} \frac{\phi_k(\tilde{x}^{(i)}_k, \tilde{x}^{(i)}_{k-1}) w_{k-1}^{(i)}}{q_k(\tilde{x}^{(i)}_k | \tilde{x}^{(i)}_{k-1}, Z_k)}, & i = 1, \ldots, L_{k-1} \\ \frac{\gamma_k(\tilde{x}^{(i)}_k)}{J_k p_k(\tilde{x}^{(i)}_k | Z_k)}, & i = L_{k-1} + 1, \ldots, L_{k-1} + J_k \end{cases} \]
Particle PHD filter: Update Step

• To perform the update step the following function is computed:

\[
\psi_{k,z}(x) = (1 - p_M(x)) g_k(z/x)
\]

• For each \( z \in Z_k \) compute

\[
C_k(z) = \sum_{j=1}^{L_{k-1}+J_k} \psi_{k,z} (\tilde{x}^{(j)}_k) \tilde{w}^{(j)}_k
\]

• For \( i=1,\ldots, L_{k-1}+J_k \), update weights as:

\[
\tilde{w}^{(i)}_k = \left[ p_M(\tilde{x}^{(i)}_k) + \sum_{z \in Z_k} \frac{\psi_{k,z} (\tilde{x}^{(i)}_k)}{\kappa_k(z) + C_k(z)} \right] \tilde{w}^{(i)}_{k\mid k-1}
\]
Particle PHD filter: Resampling Step

• The **estimated number of targets** is given by the total mass (summation of the weights)

\[
\hat{N}_{k\setminus k} = \sum_{j=1}^{L_{k-1}+J_k} \tilde{w}_k^{(j)}
\]

• A **resampling step** is applied to obtain:

\[
\begin{align*}
\left\{ \frac{\tilde{w}_k^{(i)}}{\hat{N}_{k\setminus k}}, \tilde{x}_k^{(i)} \right\}_{i=1}^{L_{k-1}+J_k} & \quad \text{to get} \quad \left\{ \frac{\tilde{w}_k^{(i)}}{\hat{N}_{k\setminus k}}, \tilde{x}_k^{(i)} \right\}_{i=1}^{L_k}
\end{align*}
\]

• Rescale the weights to get:

\[
\hat{N}_{k\setminus k} \text{ to get } \left\{ w_k^{(i)}, x_k^{(i)} \right\}_{i=1}^{L_k}
\]

• All the particles have now the **same weight**
PHD Filter

Example

Image analysis → PHD Filter → Clustering → Data Association

[Diagram showing a process flow with images of a street scene and boxes labeled Image analysis, PHD Filter, Clustering, and Data Association, with arrows connecting the boxes.]
Mathematical Models

Identity Declaration
Identity Declaration

- Identity declaration is a basic step of the processing chain of a data fusion system to associate a label to a target.
- The label can be semantic, i.e. it describes the class (e.g. vehicle vs. human) of the target, or an identification number used for associating purposes.
- Pattern Recognition techniques are employed to perform this task. For this reason usually this process is associated with feature extraction approaches in order to extract relevant information from the data.
- Many techniques can be employed in this stage (e.g. Artificial Neural Networks, Support Vector Machines, Bayesian methods, Clustering algorithms, ecc.)
Remarks on Bayesian Decision Rule

\[ R = P_0 C_{00} \int_{Z_0} p(y \mid H_0) \, dy + P_0 C_{10} \int_{Z-Z_0} p(y \mid H_0) \, dy + \]
\[ + P_1 C_{01} \int_{Z_0} p(y \mid H_0) \, dy + P_1 C_{11} \int_{Z-Z_0} p(y \mid H_1) \, dy \]

where \( Z_i \) is the decision region corresponding to hypothesis \( H_i \).

Let \( Z \) be the entire observation space so that \( Z=Z_0 \cup Z_1 \) and \( Z_0 \cap Z_1 = \emptyset \), the null set

We may expand the previous relation as:

\[ R = \sum_{i=0}^{1} \sum_{j=0}^{1} C_{ij} P_j P(\text{Decide } H_i \mid H_j) = \sum_{i=0}^{1} \sum_{j=0}^{1} C_{ij} P_j \int_{Z_i} p(y \mid H_j) \, dy \]
Remarks on Bayesian Decision Rule

• Nothing that

\[ \int_Z p(y \mid H_j) dy = 1, \quad j = 0, 1 \]

and collecting terms we can reduce to

\[ R = P_0 C_{10} + P_1 C_{11} + \]

\[ + \int_{Z_0} \left\{ P_1 (C_{01} - C_{11}) p(y \mid H_1) - P_0 (C_{10} - C_{00}) p(y \mid H_0) \right\} dy \]
Remarks on Bayesian Decision Rule

- The risk $R$ can be minimized by assigning those points of $Z$ to $Z_0$ that make the integrand negative.

- Assuming $C_{10} > C_{00}$ and $C_{01} > C_{11}$, the minimization results in the likelihood ratio test (LRT)

\[
\frac{p(y \mid H_1)}{p(y \mid H_0)} > \frac{P_0(C_{10} - C_{00})}{P_1(C_{01} - C_{11})}
\]

- The quantity on the left hand side is known as the likelihood ratio denoted by $\Lambda(y)$ and the quantity on the right hand side is the threshold $\eta$. 


Remarks on Bayesian Decision Rule

\[
\Lambda(y) = \frac{p(y \mid H_1)}{p(y \mid H_0)} \quad \eta = \frac{P_0(C_{10} - C_{00})}{P_1(C_{01} - C_{11})}
\]

- The Likelihood Ratio Test - **LRT** can then be expressed as

\[
\Lambda(y) > \eta_{H_0} \quad \eta \nless H_0
\]

thus the log Likelihood Ratio Test **logLRT** is

\[
\log \Lambda(y) > \log \eta_{H_0} \quad \log \eta_{H_0}
\]
Special case

- If the **risk coefficients** are:
  \[ C_{00} = C_{11} = 0 \]
  \[ C_{10} = C_{01} = 1 \]

- the **Bayes risk** is:
  \[ R = P_0 \int_{z_1} p(y \mid H_0) dy + P_1 \int_{z_0} p(y \mid H_1) dy \]

where \( R \) is the average probability of error

- Given that the threshold is \( \eta = \frac{P_0}{P_1} \), when \( P_0 = P_1, \eta = 1 \)

- Using this threshold the receiver has the **minimum probability of error**
Errors and associated probabilities

Probability of **false alarm**: $P_F$

Probability of **miss detection**: $P_M$

$$P_F = P(\text{Decide } H_1 \mid H_0 \text{ present}) = \int_{Z_1} p(y \mid H_0) dy$$

$$P_M = P(\text{Decide } H_0 \mid H_1 \text{ present}) = \int_{Z_0} p(y \mid H_0) dy$$

Probability of **correct detection**: $P_D$

$$P_D = 1 - P_M = \int_{Z_1} p(y \mid H_1) dy$$
Artificial Neural Networks

- Artificial Neural Networks (ANNs) consists of layers of processing elements (neurons) connected in a variety of ways that can be used for classification/recognition purposes.

- The learning process consists in assessing from a set of labeled (supervised learning) or unlabeled (unsupervised learning/clustering) examples the parameters of a mapping function \( f(\cdot) \) that associates input data \( x \) (feature vector) to the output \( y \) (class/cluster).

- In ANNs each node of the network is a function that maps a set of input into an output as:

\[
y = f \left( \sum_{j=0}^{n} w_j x_j - \theta_i \right)
\]
Artificial Neural Networks

• The function $f(\cdot)$ can assume different forms (e.g. sigmoid, step function, etc.).

• A number of variations can be used to formulate a Neural Network including the number of layer of the network, number of nodes, etc. (e.g. Multi-Layer Perceptron - MLP)

• Back-propagation is a widely used supervised training technique to find the values of the weights that minimizes an error function between the output of the network and the desired output.
Distributed Data Fusion

Distributed State Estimation
Distributed State Estimation

- The problem of **dynamic sensor fusion** handles the problem of fusing data from all the sensors (observing a phenomenon represented as a random variable \( x(t) \)) to provide an estimate.
Distributed State Estimation

• The problem can be solved if all the sensors transmit their measurements at every time step. A central node can implement a Kalman Filter on all the measurements.

• The problems of this solution are the following:
  - computational cost: the central node needs to handle matrix operations that increase in size as the number of sensors increases.
  - transmission: the sensors may not be able to transmit at every time step. Hence one may want to transmit after some local processing.

• In the following a possible way to handle the first problem (i.e. to reduce the computational burden) is proposed when the estimation is performed using a Kalman Filter.
Distributed State Estimation

Distributed Kalman Filter

• Sensors are assumed to be able to transmit information to a central node at each time step.

• Let assume that a random variable (state) evolves as:

\[ x(k+1) = Fx(k) + n(k) \]

and its measurements undergoes the equation

\[ z(k) = Hx(k) + w(k) \]

• A new form of the Kalman Filtering update is used:

\[
P^{-1}(k \mid k) \hat{x}(k \mid k) = P^{-1}(k \mid k-1) \hat{x}(k \mid k-1) + H^T R^{-1} z(k) \]

\[
P^{-1}(k \mid k) = P^{-1}(k \mid k-1) + H^T R^{-1} H \]
Distributed State Estimation

- The requirements from the **individual sensors** can be derived using the modified update Kalman Filtering equations.

- The global error covariance matrix and the estimate are given in terms of the **local covariances** and **estimates** by:

\[
P^{-1}(k \mid k) = P^{-1}(k \mid k - 1) + \sum_{i=1}^{N} \left( P^{-1}_i(k \mid k) - P^{-1}_i(k \mid k - 1) \right)
\]

\[
P^{-1}(k \mid k) \hat{x}(k \mid k) = P^{-1}(k \mid k - 1) \hat{x}(k \mid k - 1) + \sum_{i=1}^{N} \left( P^{-1}_i(k \mid k) \hat{x}_i(k \mid k) - P^{-1}_i(k \mid k - 1) \hat{x}_i(k \mid k - 1) \right)
\]  \[8\]

- On the basis of these results two possible **architectures** can be proposed for dynamic sensor fusion.
Distributed State Estimation

First Architecture

• Each individual sensor transmits its **local estimates** $\hat{x}_i(k \mid k)$. The global fusion node combines the estimates using Eq. [8].

  \[
  \hat{x}(k) = \text{Eq. [8]}
  \]

  $\hat{x}_1(k \mid k)$  $\hat{x}_2(k \mid k)$  $\hat{x}_n(k \mid k)$

• The terms $\hat{x}_i(k \mid k-1)$ and $\hat{x}(k \mid k-1)$ can be computed by the fusion node using the **time update equation**:

  \[
  \hat{x}(k \mid k-1) = F\hat{x}(k-1 \mid k-1)
  \]

• The architecture is **simple** but the fusion node has to do a lot of computation
Distributed State Estimation

Second Architecture

• The architecture is similar but instead of sending the local estimates, the sensors transmit more information to make the computation simpler for the fusion node.

• Considering that the term $P^{-1}(k \mid k-1)\hat{x}(k \mid k-1)$ can be re-written as contributions from individual sensors:

$$P^{-1}(k \mid k-1)\hat{x}(k \mid k-1) = \sum_{i=1}^{N} c_i(k)$$

• The proof of this assumption is the following:

$$P^{-1}(k \mid k-1)\hat{x}(k \mid k-1) = P^{-1}(k \mid k-1)F\hat{x}(k-1 \mid k-1) =$$

$$= P^{-1}(k \mid k-1)FP(k-1 \mid k-1)P^{-1}(k-1 \mid k-1)\hat{x}(k-1 \mid k-1) =$$

$$= P^{-1}(k \mid k-1)FP(k-1 \mid k-1)(P^{-1}(k-1 \mid k-2)\hat{x}(k-1 \mid k-2) +$$

$$+ \sum_{i=1}^{N} \left(P_i^{-1}(k-1 \mid k-1)\hat{x}_i(k-1 \mid k-1) - P_i^{-1}(k-1 \mid k-2)\hat{x}_i(k-1 \mid k-2)\right)$$
Distributed State Estimation

• Thus the contribution $c_i(k)$ evolves according to the following equation:

$$c_i(k) = P^{-1}(k \mid k-1)FP_i^{-1}(k-1 \mid k-1)c_i(k-1) +$$

$$+ \left( P_i^{-1}(k-1 \mid k-1)\hat{x}_i(k-1 \mid k-1) - P_i^{-1}(k-1 \mid k-2)\hat{x}_i(k-1 \mid k-2) \right)$$

• The covariances do not depend on the data and can be calculated anywhere. Hence each sensor transmits the quantity:

$$d_i(k) = \left( P_i^{-1}(k \mid k)\hat{x}_i(k \mid k) - P_i^{-1}(k \mid k-1)\hat{x}_i(k \mid k-1) \right) + c_i(k)$$

• The fusion node have just to compute the summation of these contributions to calculate the estimate
Distributed State Estimation

- The computation at the fusion note is far **easier** than for the first architecture.

- The **architecture** is the following, considering:

\[
d_i(k) = P_i^{-1}(k|k)\hat{x}_i(k|k) - P_i^{-1}(k|k-1)\hat{x}_i(k|k-1) + c_i(k)
\]

\[
\hat{x}(k) = \sum d_i(k)
\]

- The computation at the fusion note is far easier than for the first architecture.

- The **architecture** is the following, considering:

\[
d_i(k) = P_i^{-1}(k|k)\hat{x}_i(k|k) - P_i^{-1}(k|k-1)\hat{x}_i(k|k-1) + c_i(k)
\]

\[
\hat{x}(k) = \sum d_i(k)
\]
Distributed Data Fusion

Distributed Detection
Identity Declaration Fusion

• Identity declaration fusion consists in collecting and combining the identity declarations obtained by processing the data collected from different sensors.

• Different approaches can be found for identity fusion many of them drawing inspiration from pattern recognition (classical inference, Bayesian inference, Dempster-Shafer’s method, Fuzzy set theory, Estimation theory, Cluster analysis, Adaptive Neural Networks, ecc.).

• Many structures are also present according to where the decision processes (and features extraction) are accomplished.
A motivating example

• Let us consider to have four objects that need to be **recognized** that are characterized by two **independent features** – shape and temperature

• There are **three sensors** with different characteristics:
  ▪ sensors 1 and 2 provide **redundant** information regarding shape
  ▪ sensor 3 provides **complementary** information regarding temperature

From [B5]
A motivating example

• Fusion of redundant information usually reduces uncertainty

• **Problem** of sensor 1 and 2 fusion: cannot distinguish objects of the same shape

From [B5]
A motivating example

- The figure shows that the fusion of **complementary information** provides the ability for **discrimination**
- In fact object A, C and B, D can’t be distinguished just by using sensors 1 and 2 that provide **redundant information** on the shape feature
- Sensor 3 enables the discrimination proving **complementary information** regarding the temperature
Parallel Fusion Networks without Fusion

• In this situation local decisions are **not fused** and they **do not communicate** with each other.

• However their operation is coupled since the **costs** of decision making are coupled and a system-wide optimization is performed.

> A two detector parallel fusion network without fusion.

From [B5]
Parallel Fusion Networks without Fusion

- Considering the **observations** at the two detectors as \( y_1 \) and \( y_2 \) and the **decisions** as \( u_1 \) and \( u_2 \) it is possible to write the Bayesian risk as:

\[
\mathcal{R} = \sum_{i,j,k} \int_{y_1, y_2} p(u_1, u_2, y_1, y_2, H_k) C_{i,j,k}
\]

where \( C_{ijk}, \ i,j,k = 0,1 \) is the cost of detector 1 deciding \( H_i \), of detector 2 deciding \( H_j \) when \( H_k \) is present.

- Assuming the **independence** between observations it is possible to write the **likelihood ratio at detector 1** (that is analogous to the one at detector 2) as:

\[
\Lambda(y_1) = \frac{p(y_1 \mid H_1)}{p(y_1 \mid H_0)}
\]

\[
\begin{align*}
&= \begin{cases} 
  \sum_{j} \int_{y_2} p(u_2 \mid y_2) p(y_2 \mid y_1, H_0) & u_{t=1} \\
  \sum_{j} \int_{y_2} p(u_2 \mid y_2) p(y_2 \mid y_1, H_1) & u_{t=0}
\end{cases} \\
&= \frac{P_0 \sum_j \int p(u_2 \mid y_2) p(y_2 \mid y_1, H_0) [C_{1j0} - C_{0j0}]}{P_1 \sum_j \int p(u_2 \mid y_2) p(y_2 \mid y_1, H_1) [C_{1j1} - C_{0j1}]}
\end{align*}
\]
Design of fusion rules

• Local decision can be then integrated into a data fusion center by using a **fusion rule**

• Let us consider the inputs to the fusion center: $u_i, i=1,...,N$ and the output of the fusion center: $u_0$.

• The **Fusion rule** is logical function with N binary inputs and one binary output. In general there are $2^{2^N}$ possible fusion rules.

Local decision

$$u_i = \begin{cases} 
0 & \text{if detector } i \text{ decides } H_0 \\
1 & \text{otherwise}
\end{cases}$$

Global decision

$$u_0 = \begin{cases} 
0 & \text{if } H_0 \text{ is decided} \\
1 & \text{otherwise}
\end{cases}$$
Design of fusion rules

- Possible **Fusion rules** for two detectors binary decisions are:

- An *arbitrary* choice of one of the fusion rule (e.g. AND, OR, MAJORITY) is usually performed.

- Some of the possible rules may **not be preferred** for many fusion applications, e.g. $f_1$ and $f_{16}$ (all 0 and 1) or the ones that disregard one of the two decisions (e.g. $f_4$ disregards $u_2$).

![Table showing possible fusion rules](image)

From [B5]
Design of fusion rules

- **Logical functions** are usually the more suitable:

**AND rule**

\[ u_0 = \begin{cases} 
1, & \text{if } u_1 = 1 \text{ and } u_2 = 1 \\
0, & \text{otherwise} 
\end{cases} \]

**OR rule**

\[ u_0 = \begin{cases} 
1, & \text{if } u_1 = 0 \text{ and } u_2 = 0 \\
0, & \text{otherwise} 
\end{cases} \]

- In the following the **fusion rule** is derived in the **Bayesian Framework** to find an optimal solution (according to the minimization of the average cost criterion).
Design of fusion rules

• It is possible to define the probabilities of false alarm, detection and miss detection for the i-th detector

\[
\begin{align*}
P_{Fi} &= P(u_i = 1 \mid H_0) \\
P_{Di} &= P(u_i = 1 \mid H_1) \\
P_{Mi} &= P(u_i = 0 \mid H_1)
\end{align*}
\]

• The **optimum fusion rule** is given by the LRT

\[
\begin{align*}
P(u_1, u_2, \ldots, u_n \mid H_1)_{u_0=1} &> P_0(C_{10} - C_{00}) \\
P(u_1, u_2, \ldots, u_n \mid H_0)_{u_0=0} &< P_1(C_{01} - C_{11})
\end{align*}
\]

where \( C_{ij} \) denotes the cost of global decision being \( H_i \), when \( H_j \) is present.
Design of fusion rules

• Due to the **independence of local decisions**, the left hand side of the previous relation can be written as

\[
\frac{P(u_1, u_2, \ldots, u_n \mid H_1)}{P(u_1, u_2, \ldots, u_n \mid H_0)} = \prod_{i=1}^{N} \frac{P(u_i \mid H_1)}{P(u_i \mid H_0)} = \prod_{S_1} \frac{P(u_i = 1 \mid H_1)}{P(u_i = 1 \mid H_0)} = \prod_{S_0} \frac{P(u_i = 0 \mid H_1)}{P(u_i = 0 \mid H_0)} = \prod_{S_1} \frac{1 - P_{Mi}}{P_{Fi}} \prod_{S_0} \frac{P_{Mi}}{1 - P_{Fi}}
\]

where \( S_j \) is the set of all those **local decisions** that are equal to \( j, j=0,1 \). Substituting and taking the logarithm of both sides

\[
\sum_{S_1} \log \frac{1 - P_{Mi}}{P_{Fi}} + \sum_{S_0} \log \frac{P_{Mi}}{1 - P_{Fi}} \stackrel{u_0=1}{<} \log \eta \quad \stackrel{u_0=0}{>}
\]
Design of fusion rules

• The optimum fusion rule can be implemented by forming a weighted sum of the incoming **local decisions** and, then, comparing it with a threshold (rewriting Eq. 1):

\[
\sum_{i=1}^{N} \left[ u_i \log \frac{1 - P_{Mi}}{P_{Fi}} + (1 - u_i) \log \frac{P_{Mi}}{1 - P_{Fi}} \right]_{u_0=1} > \log \eta \\
\sum_{i=1}^{N} \left[ \log \frac{(1 - P_{Mi})(1 - P_{Fi})}{P_{Mi}P_{Fi}} \right]_{u_0=1} > \log \left[ \eta \prod_{i=1}^{N} \frac{1 - P_{Fi}}{P_{Mi}} \right]_{u_0=0}
\]

• Rules can be inferred **analyzing the disequation** to derive the result of each combination of local decisions.

• The **weights** and the **threshold** are determined by the reliability of the decision, i.e. by the probability of miss and false alarm of local detectors.
Design of fusion rules

• The optimum fusion rule is **monotonic** under the assumption that $P_{D_i} \geq P_{f_i}$, $i=1,\ldots, N$. A fusion rule is monotonic if considering a set of $k$ local decision $S_1(k)$ and $N-k$ local decision $S_0(N-k)$ that produce an output $u_0=1$, it must give $u_0=1$ for every $S_1(k')$ with $k'>k$ and such that $S_1(k')$ contains $S_1(k)$.

• For example let’s consider $S_1(k)$: $u_1=1; u_3=1; u_5=1$ and $S_0(k)$: $u_2=0; u_4=0$ and the fusion rule that give $u_0=1$ with these local decisions. The **fusion rule is monotonic** if it gives 1 also for $S_1(k')$: $u_1=1; u_2=1; u_3=1; u_5=1$, $S_1(k'')$: $u_1=1; u_3=1; u_4=1; u_5=1$ and $S_1(k''')$: $u_1=1; u_2=1; u_3=1; u_4=1; u_5=1$. 

Example 1: Fusion of two detectors

- Considering two local decisions and the probabilities of false alarm and miss detection:

\[ P_{fi} = 0.2 \quad P_{mi} = 0.1 \quad i=1,2 \quad C_{00} = C_{11} = 0 \quad \text{and} \quad C_{01} = C_{10} = 1 \]

- The monotonic fusion rules are \( f_1, f_2, f_4, f_6, f_8, f_{16} \). The optimum fusion rule can be determined from the Eq. (9) examined different values of \( P_0 \) as shown in the following table.

<table>
<thead>
<tr>
<th>Range of values for ( P_0 )</th>
<th>Optimum fusion rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - 0.0154</td>
<td>( f_{16} ) (all ones)</td>
</tr>
<tr>
<td>0.0154 - 0.36</td>
<td>( f_8 ) (OR)</td>
</tr>
<tr>
<td>0.36 - 0.953</td>
<td>( f_2 ) (AND)</td>
</tr>
<tr>
<td>0.953 - 1</td>
<td>( f_1 ) (all zeros)</td>
</tr>
</tbody>
</table>

From [B5]
Design of fusion rules

Example 2: Fusion of two detectors

- If the values of the probabilities of false alarm and miss detection changes as:

\[ P_{Fi} = P_{Mi} = 0.1, \quad C_{00} = C_{11} = 0, \quad C_{01} = C_{10} = 1 \]

then the range of \( P_0 \) to evaluate the optimal fusion rule changes as shown in the following table:

<table>
<thead>
<tr>
<th>Range of values for ( P_0 )</th>
<th>Optimum fusion rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 – 0.012</td>
<td>( f_{16} ) (all ones)</td>
</tr>
<tr>
<td>0.012 – 0.5</td>
<td>( f_B ) (OR)</td>
</tr>
<tr>
<td>0.5 – 0.988</td>
<td>( f_2 ) (AND)</td>
</tr>
<tr>
<td>0.988 – 1</td>
<td>( f_1 ) (all zeros)</td>
</tr>
</tbody>
</table>

From [B5]
Design of fusion rules

Example: Fusion of N sensors decisions

- Fusion rule can be expressed as a function of the number of sensors $N$
  
  $P_{fi} = 0.2 \quad P_{mi} = 0.1 \quad P_0 = 0.5$

- Therefore the decision fusion performance can be determined with respect to a function of number of sensors

From [B5]
Detection with Parallel Fusion Network

- The detection in Parallel Fusion Network has to take into account the detection by **distributed detectors** and the fusion of **local decisions**.

- In Parallel Fusion Networks each detector (DM) **makes a decision** and transmit it to the fusion node where a fusion rule combines the local decision to provide an output.

- There is the need to design the **decision rules at the peripheral detectors** and the **fusion rule** that jointly minimize the average cost.

From [B5]
Decision rules for Parallel Networks

• The derivation of a fusion rule for a Parallel Network is a difficult optimization problem.

• One approach is person-by-person optimization (PBPO). The decentralized detection system is viewed as one team member, whereas the aggregation of the local detector is the other team member. The second team can be further be seen as a team consisting of local detectors as team members.

• While optimizing one team member the others are considered already designed and remain fixed.

• The number of possible fusion rules is $2^N$ since it combines $N$ local decisions.
Decision rules for Parallel Networks

- The **local decision rule** can be expressed as:

\[
p(y_k \mid H_1) = \begin{cases} 
  \frac{\sum u^k C_F A(u^k)}{\sum u^k C_D A(u^k)} & u_k = 1 \\
  \frac{\sum u^k C_D A(u^k)}{\sum u^k C_F A(u^k)} & u_k = 0
\end{cases}
\]

where:

\[
C_F = P_0 (C_{10} - C_{00}), \quad C_D = (1 - P_0) (C_{01} - C_{11})
\]

\[
A(u^k) = P(u_0 = 1 \mid u^{k1}) - P(u_0 = 1 \mid u^{k0})
\]

\[
u^k = (u_1, ..., u_{k-1}, u_{k+1}, ..., u_k)^T
\]

\[
u^{kj} = (u_1, ..., u_k = j, ..., u_k)^T \quad j = 0, 1
\]
Decision rules for Parallel Networks

- The PBPO gives the **fusion rule** $u^*$ that can be expressed as:

$$
\frac{P(u^* \mid H_1)}{P(u^* \mid H_0)}
\begin{cases}
  > C_F & u_0 = 1 \\
  < C_D & u_0 = 0
\end{cases}
$$

- In the special case of **conditionally independent local observations** it is possible to write the local decision as:

$$
p(y_k \mid H_1) p(y_k \mid H_0)
\begin{cases}
  > \sum_{u^k} C_F A(u^k) \prod_{i=1, i \neq k}^N P(u_i \mid H_0) & u_k = 1 \\
  < \sum_{u^k} C_D A(u^k) \prod_{i=1, i \neq k}^N P(u_i \mid H_1) & u_k = 0
\end{cases}
$$

and the **fusion rule** as:

$$
\prod_{i=1}^N \frac{P(u_i \mid H_1)}{P(u_i \mid H_0)}
\begin{cases}
  > C_F & u_0 = 1 \\
  < C_D & u_0 = 0
\end{cases}
$$
Decision rules for Parallel Networks

Example: Fusion of two detectors in a Parallel Network

• Let us assume to have a two detectors parallel fusion networks with conditionally independent observations, Gaussian conditional densities and minimum probability of error cost assignment (i.e. $C_{00}=C_{11}=0$; $C_{01}=C_{10}=1$).

• The fusion rule can be derived from Eq. 3 (see also pag. 115) and is given by the following four relationships:

$$
\begin{align*}
\frac{P_{M1}}{1-P_{F1}} & > \frac{P_{M2}}{1-P_{F2}} & & u_0=1 \\
\frac{P_{M1}}{P_{F1}} & < \frac{P_{M2}}{1-P_{F2}} & & u_0=0 \\
\frac{1-P_{M1}}{1-P_{F1}} & > \frac{P_{M2}}{1-P_{F2}} & & u_0=1 \\
\frac{1-P_{M1}}{P_{F1}} & < \frac{P_{M2}}{1-P_{F2}} & & u_0=0
\end{align*}
$$
Decision rules for Parallel Networks

• The **two thresholds** for each detector computed according to Eq. 10, for example for detector 1 we have:

\[
 t_1 = \frac{\sum_{u_2} C_F A(u_2) P(u_2 | H_0)}{\sum_{u_2} C_D A(u_2) P(u_2 | H_1)} = \frac{C_F}{C_D} P_{F2} \left( P_{100} - P_{110} + P_{111} - P_{101} \right) + \left( P_{110} - P_{100} \right) \]

where

\[
P_{ijk} = P(u_0 = i | u_1 = j, u_2 = k)
\]

• The simultaneous solution of the four fusion rules and the two thresholds yields the **result**.
Decision rules for Parallel Networks

• Let us consider the conditional density as **Gaussian distributions** where under $H_0$, they are identical densities with mean zero and variance one and under $H_1$, with mean $m_1$ and $m_2$, variance one.

• The resulting **fusion rules** derived from Eq. 10 and 11 are expressed in function of $m_1$ and $m_2$ and $P_0$.

<table>
<thead>
<tr>
<th>$m_1$, $m_2$</th>
<th>Range of values for $P_0$</th>
<th>Optimum fusion rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m_1 = 1$</td>
<td>0 - 0.5</td>
<td>$f_8$ (OR)</td>
</tr>
<tr>
<td>$m_2 = 1$</td>
<td>0.5 - 1.0</td>
<td>$f_2$ (AND)</td>
</tr>
<tr>
<td>$m_1 = 1$</td>
<td>0 - 0.185</td>
<td>$f_6$ (same as DM 2)</td>
</tr>
<tr>
<td>$m_2 = 1.5$</td>
<td>0.185 - 0.5</td>
<td>$f_8$ (OR)</td>
</tr>
<tr>
<td></td>
<td>0.5 - 0.815</td>
<td>$f_2$ (AND)</td>
</tr>
<tr>
<td></td>
<td>0.815 - 1.0</td>
<td>$f_6$ (same as DM 2)</td>
</tr>
<tr>
<td>$m_1 = 1$</td>
<td>0 - 0.38</td>
<td>$f_6$ (same as DM 2)</td>
</tr>
<tr>
<td>$m_2 = 2$</td>
<td>0.38 - 0.5</td>
<td>$f_8$ (OR)</td>
</tr>
<tr>
<td></td>
<td>0.5 - 0.62</td>
<td>$f_2$ (AND)</td>
</tr>
<tr>
<td></td>
<td>0.62 - 1.0</td>
<td>$f_6$ (same as DM 2)</td>
</tr>
</tbody>
</table>

From [B5]
Detection with Parallel Fusion Network

• The figure shows the **thresholds** at the two detectors for different value of $P_0$, considering $m_1 = m_2 = 1$

• In this case the thresholds for the two detectors have **the same** value

• The **discontinuities** in the threshold value is due to the use of a different rule

From [B5]
• The figure shows the **resulting probability of error** for different value of $P_0$, considering $m_1 = m_2 = 1$,

• The probability of error for the **AND** and **OR** fusion rules is shown for different value of $P_0$
Bayesian networks – Representation

- Our goal is to **represent the joint distribution** over a set of random variables:

\[ \mathcal{X} = \{ X_1, \ldots, X_n \} \]

- Let us define \( \text{Val}(X_1) \) as all the possible discrete assignments of \( X_1 \)

- The probability space of the **full joint distribution** is

\[ \prod_{i=1}^{n} \text{val}(X_i) \]  
(Eq. 1)
Bayesian networks – Simplest Example

- Consider the problem of a company trying to hire a recent college graduate. **The goal is to hire intelligent students** but there is **not possibility to measure intelligence directly**.
- The company has access to the Scholastic Aptitude Test (SAT) scores, which are informative but not fully indicative.
- In this simple example, we induce two random variables: **Intelligence** that can take values \{i^0, i^1\} and **SAT** score that can take values \{s^0, s^1\}.
Bayesian networks – Simplest Example

• In this case that our random variables are discrete and binary-valued, the number of non-redundant parameters for a full distribution is defined by $2^n - 1$ (the last parameter is fully defined by the others) equal to $2^2 - 1 = 3$

<table>
<thead>
<tr>
<th>$i^0$</th>
<th>Low Intelligence</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i^1$</td>
<td>High Intelligence</td>
</tr>
<tr>
<td>$s^0$</td>
<td>Low score</td>
</tr>
<tr>
<td>$s^1$</td>
<td>High score</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$I$</th>
<th>$S$</th>
<th>$P(I,S)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i^0$</td>
<td>$s^0$</td>
<td>0.665</td>
</tr>
<tr>
<td>$i^0$</td>
<td>$s^1$</td>
<td>0.035</td>
</tr>
<tr>
<td>$i^1$</td>
<td>$s^0$</td>
<td>0.060</td>
</tr>
<tr>
<td>$i^1$</td>
<td>$s^1$</td>
<td>0.024</td>
</tr>
</tbody>
</table>
Bayesian networks – Simplest Example

• **Factorizing** the joint distribution $p(I,S)$ can give us more natural causality meaning to the parameters.

• Our probability distribution can be factorized as

$$p(I,S) = p(I)p(S | I)$$

• And the original distribution $p(I,S)$ can be represented with the following CPTs:

<table>
<thead>
<tr>
<th>I</th>
<th>s^0</th>
<th>s^1</th>
</tr>
</thead>
<tbody>
<tr>
<td>i^0</td>
<td>0.95</td>
<td>0.05</td>
</tr>
<tr>
<td>i^1</td>
<td>0.20</td>
<td>0.80</td>
</tr>
</tbody>
</table>
Bayesian networks – Simplest Example

- From the mathematical perspective, the last alternative leads to the exactly same joint distribution (you can prove yourself by doing the math).

- The 2 defined CPTs required 3 non-redundant parameters to be fully specified. 1 parameter from the binomial distribution from first table and 2 parameters for the two binomial distributions of second table (one for every possible assignment of the parents).
• Now let’s assume that the company has also access to the student’s grade in some course. **We can enhance our Bayesian Network with the variable** \( G \) **that can take the values** \( \{g^1, g^2, g^3\} \)

\[
p(I, S, G) = p(I)p(S|I)p(G|I)
\]
Some important remarks about the enhanced model:

- Although all the joint distribution changed, the CPTs defined in the first example are still valid. *(adding nodes to some parts of the graphs does not mean change all the graph)*
- According to Eq.(1), the total number of non-redundant parameters to describe the full distribution is 11.
- Using the current factorization allows to describe the distribution with only 7 non-redundant parameters. Which mean that the factorized distribution is more compact.
- The less number of parameter come from the missing link between $G$ and $S$. Implicitly this implies that Grades is conditional independent from SAT given Intelligence.
Bayesian networks – Third Example

- Lets consider a more complex scenario where we add two random variables $D$ and $L$
  - The grades now depend on how difficult the course is. It can take the values $\{d^0, d^1\} = \{\text{easy, hard}\}$
  - And a letter recommendation that can take values $\{d^0, d^1\} = \{\text{strong, weak}\}$
Bayesian networks – Third Example

• Difficulty

• SAT

• Intelligence

• Grade

• Letter

\[
\begin{array}{ccc}
  d^0 & d^1 \\
  0.6 & 0.4 \\
\end{array}
\]

\[
\begin{array}{ccc}
  i^0 & i^1 \\
  0.7 & 0.3 \\
\end{array}
\]

\[
\begin{array}{ccc}
  g^1 & g^2 & g^3 \\
  i^0 & d^0 & 0.30 & 0.40 & 0.30 \\
  i^0 & d^1 & 0.05 & 0.25 & 0.70 \\
  i^1 & d^0 & 0.90 & 0.08 & 0.02 \\
  i^1 & d^1 & 0.50 & 0.30 & 0.20 \\
  j^0 & s^0 & 0.95 & 0.05 \\
  j^1 & s^1 & 0.20 & 0.80 \\
\end{array}
\]

\[
\begin{array}{cc}
  l^0 & l^1 \\
  g^0 & 0.10 & 0.90 \\
  g^1 & 0.40 & 0.60 \\
  g^2 & 0.99 & 0.01 \\
\end{array}
\]
Bayesian networks – Third Example

- What is the **factorization** of the presented graph?
- What is the **probability space dimension** of the **full probability distribution**?
- What is the total number of **non-redundant parameters** of the **full distribution**?
- What is the total number of **non-redundant parameters** of the **factorized distribution**?
Bayesian networks – Third Example

• What is the factorization of the presented graph?
  \[ p(I, D, G, S, L) = p(I)p(D)p(G|I, D)p(S|I)p(L|G) \]

• What is the probability space dimension of the full probability distribution?
  • 48

• What is the total number of non-redundant parameters of the full posterior?
  • 47

• What is the total number of non-redundant parameters of the factorized distribution?
  • 11
Bayesian networks – Third Example

• How to calculate entrances of the full distribution using the CPTs?

• Let’s say we want to find the joint probability of:
  ▪ A student being intelligent.
  ▪ The course to be easy.
  ▪ To obtain a “B” in given course.
  ▪ Obtain a good score in SAT exam.
  ▪ Receive a strong recommendation letter.

\[
p(i^1, d^0, g^2, s^1, l^0) = p(i^1) p(d^0) p(g^2 | i^1, d^0) p(s^1 | i^1) p(l^0 | g^2)
\]

\[
= 0.3 * 0.6 * 0.08 * 0.8 * 0.4 = 0.004608
\]
Bayesian networks – Behavior with Evidence

• Once selected the random variables, constructed the Graphical Model and set the CPTs, **we can infer the posterior probability of an event given some evidence.**

  \[ p(Y = y \mid E = e) \]

• **The naive way** obtain the this posterior probability is by eliminating the entries in the joint inconsistent with our observation “e” and renormalize the result entries to sum up to “1”. Then we compute the probability of the event “y” by summing the probabilities of all of the entries in the resulting posterior that are consistent with “y”.
Bayesian networks – Behavior with Evidence

- Let’s see how the probabilities change once we get evidence.
  - The probability to obtain a strong recommendation without any evidence is around 50.2% \( p(l^1) \approx 0.502 \)
  - If we know that the student is not intelligent, this probability decreases to 38.9% \( p(l^1 | i^0) \approx 0.389 \)
  - If we discover that the course is an easy class, the probability increases again to 51.3% \( p(l^1 | i^0, d^0) \approx 0.513 \)
Let’s see another interesting example of the effect called “explaining away”.

- Without seeing any evidence, our belief that a student is intelligent is 30%: $p(i^1) = 0.300$
- If we have the evidence that the student got a C in the course, the probability of being intelligent decreases to 7.9% but at the same time the probability of the course being difficult increases from 40% to 62.9%:
  \[
  p(i^1 | g^3) \approx 0.079 \quad \text{and} \quad p(d^1 | g^3) \approx 0.629
  \]
- If the student submits the SAT score with a high score, his probability of being intelligent goes from 7.9% to 57.8% and the probability of the course to be difficult to 76%:
  \[
  p(i^1 | g^3, s^1) \approx 0.578 \quad \text{and} \quad p(d^1 | g^3, s^1) \approx 0.760
  \]
Bayesian networks – Behavior with Evidence

- In the last example the high SAT score outweighs the poor grade because low intelligence students are extremely unlikely to receive high scores in SAT, whereas high intelligence students can still get C’s if the course is difficult.

- *Explaining away* is an instance of a general reasoning pattern called *intercasual reasoning*, where different causes of the same effect can interact. This type of reasoning is a very common pattern in human reasoning.
Bayesian networks – Graph Independences

- What are the independence that can be drawn from the graph?

\( (L \perp I, D, S \mid G) \)

\( (S \perp D, G, L \mid I) \)

\( (G \perp S \mid I, D) \)

\( (I \perp D) \)

\( (D \perp I, S) \)
Dynamic Bayesian networks

- **Dynamic Bayesian Networks (DBNs)** can be considered as an extension of Bayesian Networks to handle temporal models.
- The term “dynamic” is due to the fact that they are used to represent a **dynamic model** (A model with a variable state over time).
- A **DBN is defined** by $(B_0, B \rightarrow)$ where $B_0$ defines the prior probability over the state and $B \rightarrow$ is a two-slice temporal Bayes net (2TBN) which defines how the systems evolves in time.
Dynamic Bayesian networks

- There are two types of edges (dependencies) that can be defined in a DBN. **Intra-slice** topology (within a slice) and **inter-slice** topology (between two slices).
Dynamic Bayesian networks

- The decision of **how to relate two variables**, if either intra-slice (aka intra-time-slice) or inter-slice (aka inter-time-slice) depends on how tight the coupling is between them.
- If the **effect** of one variable on the other is **immediate** (much shorter than the time granularity) the influence should manifest as **intra-slice** edge.
- If the **effect** is slightly **longer-term** the influence should manifest as **inter-slice** edge.
- An inter-slice edge connecting two instances of the same variable is called **persistence-edge**.
Dynamic Bayesian networks

- The **DBN** structure **must satisfy** the following assumptions:
  - The **structure and CPDs** in the do not change over time.
  - **Inter-slice** arcs are all **from left to right**, in accordance with the temporal evolution.
  - **No cycles** must be present in the **intra-slice** arcs.
- Thus we can view a **DBN** as a compact representation from which we **can generate an infinite** set of **Bayesian networks** (one for every T>0)
Dynamic Bayesian networks

- Hidden Markov Models (HMMs) and Kalman Filter Model (KFM) are specific nontrivial examples of DBNs.
- They are formed by one hidden variable with persistence links between time steps and one observed.
Dynamic Bayesian networks – HMM

- HHM is characterized by one discrete hidden node.
- The probabilities that have to be defined are:
  - $p(x_0)$ that is the **initial state distribution** and represents the uncertainty on the initial value of the state.
  - $p(x_k | x_{k-1})$ that is the **transition model**. It describes how the state evolves in time.
  - $p(z_k | x_k)$ that is the **observation model** and represents how the observations are related and generated by the hidden state. It is also called **likelihood**.
Dynamic Bayesian networks – KFM Revisited

- KFM is characterized by one **continuous hidden node**.
- All nodes are assumed to be **linear-Gaussian distributions**.
- The probabilities then defined as:
  - **Initial state**
    \[ p(x_0) = N(x_0, Q_0) \]
  - **Transition model**
    \[ p(x_k | x_{k-1}) = N(Fx_{k-1} + Gu_k, Q) \]
  - **Observation model**
    \[ p(z_k | x_k) = N(Hx, V) \]
Dynamic Bayesian networks – Data Fusion

- There mainly three ways to fuse observations in DBNs

- Conditionally independent fusion
- Linearly conditionally dependent fusion
- Conditionally dependent fusion
• **Mathematically** this relations can be expressed, defining \( Z_k = \{z_{1k}^1, z_{2k}^2, \ldots, z_{L_k}^L\} \) as the set of different observations (or sensors), as:
  
  - **Conditionally indepent fusion**
    
    \[
p(Z_k | x_k) = p(z_{1k}^1 | x_k)p(z_{2k}^2 | x_k)\ldots p(z_{L_k}^L | x_k)
    \]
  
  - **Linearly conditionally dependent fusion**
    
    \[
p(Z_k | x_k) = \alpha_{1k}^1 p(z_{1k}^1 | x_k) + \alpha_{2k}^2 p(z_{2k}^2 | x_k) + \ldots + \alpha_{L_k}^L p(z_{L_k}^L | x_k)
    \]
    
    Subject to: \( \sum_{i} \alpha_{ik} = 1 \)
  
  - **Conditionally dependent fusion**
    
    \[
p(Z_k | x_k) = p(z_{L_k}^L | z_{1:L-1_k}^1, x_k)p(z_{L-1_k}^{L-1} | z_{1:L-2_k}^1, x_k)\ldots p(z_{1k}^1 | x_k)
    \]
Dynamic Bayesian networks – Data Fusion

• Let’s consider the example of visual tracking:
  ▪ The likelihood coming from motion and color can be taken as conditionally independent (the motion of the object can be assumed not correlated to its motion)
  ▪ The problem of fusing different cues in order to create active discriminative appearance models using different color spaces can be fused with conditional linear dependency where more weight is given to the cue that is more discriminative at that time step.
  ▪ Now consider the case where we not only want to use different color spaces but we want to actively find the color space that best separates foreground and background. Then try to find the best color description in this color space. In this case, the color description depends on the color space chosen.
Dynamic Bayesian networks - Advance DBN structures

- The DBN in the figure is constructed from HMMs and is called *factorial HMM*.
- This type of model is *very useful* in a variety of applications, for example, when *several sources* of sound are being heard simultaneously through a *single microphone*.
The DBN in the figure is called **coupled HMM**.

This type of model is constructed from a set of chains, with **each chain having its own observation**. Chains **interact** via their state variable affecting adjacent chains.

This kind of HMM is useful, for example, to **model interaction** between different interacting objects.
Markov random field: illustrative example

- Exploiting dependencies using *Graph cuts* algorithm on MRF


References

